python™

Projeto Final

VERSÁTIL
tecnologia em informação

Sumário

SEPN 509 BL D 1º ANDAR ED. ISIS - ASA NORTE - BRASÍLIA/DF
☎ (61) 3340 1815 | 3447 7520 | 97401 3340
www.versatildf.com.br

1. Resolvendo as dependências do projeto.

Usando o comando ***pip install*** no **prompt de comando** instale as seguintes as seguintes bibliotecas**.**

- **PySimpleGUI**

  Para o uso da interface gráfica.

  ```
  pip install PySimpleGUI
  ```

- **datetime**

  Para o uso de data e hora.

  ```
  pip install datetime
  ```

- **random**

  Para geração aleatória de valores para o gráfico.

  ```
  pip install random
  ```

- **requests**

  Para consumo da api de previsão do tempo.

  ```
  pip install requests
  ```

- **locale**

  Para localização atual.

  ```
  pip install locale
  ```

- **mysql.connector**

  Para conexão com o banco de dados.

  ```
  pip install mysql.connector
  ```


SEPN 509 BL D 1º ANDAR ED. ISIS - ASA NORTE - BRASÍLIA/DF
☎ (61) 3340 1815 | 3447 7520 | 97401 3340
www.versatildf.com.br

Após as instalações procedemos com as importações conforme a figura abaixo:

```
import PySimpleGUI as sg
from datetime import date
import random
import requests
import locale as l
import mysql.connector as mysql
```

2. Criando a conexão com banco de dados.

2.1 função para conexão

Nessa etapa criamos variáveis para receber valores vindos dos inputs.

```
def banco():
    # variáves recebem valores dos inputs
    nome = values['-nome-']
    email = values['-email-']
    telefone = values['-telefone-']
```

2.1.1 Usando o bloco try

A estrutura **try** serve para executar um bloco de comandos, a princípio os comandos usados na conexão com o banco de dados.

```
    try:
        # esta é a função de conexão com o banco
        # devem ser fornecidos as configurações de conexão
        conexao = mysql.connect(
            host="127.0.0.1", # ip do servidor
            user="root", # usuario
            password="root", # senha do usuario
            database="dbpython" # base de dados
        )
```


SEPN 509 BL D 1º ANDAR ED. ISIS - ASA NORTE - BRASÍLIA/DF
☎ (61) 3340 1815 | 3447 7520 | 97401 3340
www.versatildf.com.br

Seguimos com mensagens para conferir o status da conexão.

```
    # mensagem para verificação da conexão
    print("Conexão realizada com sucesso.")
    # abre um Cursor para executar um SQL
    cursor = conexao.cursor()
    # comando SQL a ser executado
```

Criamos a consulta SQL para inserir as informações dos inputs para o banco de dados.

```
    sql = "INSERT INTO contatos(nome, email, telefone) VALUES (%s, %s, %s)"
    vals = (nome, email, telefone) # devem ser passados como tupla
    # executando o comando pelo Cursor
    cursor.execute(sql, vals)
    # efetivando a alteração
    conexao.commit() # obrigatório para INSERT, DELETE e UPDATE
    print("Salvo com sucesso.")
```

2.1.2 Tratamento de exceções

Por fim tratamos as exceções.

```
except mysql.Error as e:
    # capturando possiveis erros de conexao ou SQL com TRY CATCH
    print(e.msg)
```

3. Criando gráfico de demonstração.

3.1 Função para criação do layout do gráfico

Nessa etapa criamos uma função com variáveis para construir a parte visual do gráfico.

```
# função para disparar pop-up do gráfico, no evento click do botão Graf
def grafico():
    BAR_WIDTH = 50  # largura de cada barra
    BAR_SPACING = 75  # espaço entre as barras
    EDGE_OFFSET = 3  # offset a esquerda da primeira barra
    GRAPH_SIZE = DATA_SIZE = (300, 400)  # tamanho do pop-up em pixels
    # tema, o mesmo definido para o Dashboard
```


SEPN 509 BL D 1º ANDAR ED. ISIS - ASA NORTE - BRASÍLIA/DF
☎ (61) 3340 1815 | 3447 7520 | 97401 3340
www.versatildf.com.br

Em seguida determinamos o tema, definimos o layout e a janela.

```
sg.theme('Dashboard')
# layout do pop-up
layout = [[sg.Text('Gráfico de barras com PySimpleGUI')],
          [sg.Graph(GRAPH_SIZE, (0, 0), DATA_SIZE, k='-GRAPH-')],
          [sg.Button('OK'), sg.T('Click para ver mais dados'), sg.Exit()]]
# janela do pop-up
window = sg.Window('Gráfico de barras', layout, finalize=True)
```

Para finalizar essa etapa, faremos a área de plotagem, com os eixos e as colunas de dados, usamos uma estrutura de repetição para que cada elemento representado seja configurado.

```
graph = window['-GRAPH-']  # type: sg.Graph
# estrutura de repetição para geração de dados aleatórias
while True:
    # estrutura de repetição para desenho das barras, após apagar o gráfico anterior
    graph.erase()
    for i in range(7):
        graph_value = random.randint(0, GRAPH_SIZE[1])
        graph.draw_rectangle(top_left=(i * BAR_SPACING + EDGE_OFFSET, graph_value),
                             bottom_right=(i * BAR_SPACING + EDGE_OFFSET + BAR_WIDTH, 0),
                             fill_color=sg.theme_button_color()[1])
        graph.draw_text(text=graph_value, location=(i * BAR_SPACING + EDGE_OFFSET + 25, graph_value + 10))
    # normalmente o topo do loop, Mas por conta do tipo de desenho vem o gráfico antes, marque isso no botão
    event, values = window.read()
    # estrutura de decisão para definir ações dos botões
    if event in (sg.WIN_CLOSED, 'Exit'):
        break
# fim para instruções da função
window.close()
```

4. Dicionário tema.

Usado para diagramar a tela do dashboard.

```
# dicionário de temas para o dashboard
theme_dict = {'BACKGROUND': '#2B475D',
              'TEXT': '#FFFFFF',
              'INPUT': '#F2EFE8',
              'TEXT_INPUT': '#000000',
              'SCROLL': '#F2EFE8',
              'BUTTON': ('#000000', '#C2D4D8'),
              'PROGRESS': ('#FFFFFF', '#C7D5E0'),
              'BORDER': 1, 'SLIDER_DEPTH': 0, 'PROGRESS_DEPTH': 0}
# sg.theme_adiciona um tema('Dashboard', theme_dict)
sg.LOOK_AND_FEEL_TABLE['Dashboard'] = theme_dict
sg.theme('Dashboard')
# posição, borda e cores
BORDER_COLOR = '#C7D5E0'
DARK_HEADER_COLOR = '#1B2838'
BPAD_TOP = ((20, 20), (20, 10))
BPAD_LEFT = ((20, 10), (0, 10))
BPAD_LEFT_INSIDE = (0, 10)
BPAD_RIGHT = ((10, 20), (10, 20))
```


SEPN 509 BL D 1º ANDAR ED. ISIS - ASA NORTE - BRASÍLIA/DF
(61) 3340 1815 | 3447 7520 | 97401 3340
www.versatildf.com.br

5. Data e hora

Para definir data e hora, precisamos definir a localidade, converter a data para o formato de texto, visando formatar de acordo com o formato que usamos no Brasil, por extenso e determinar como ficará o banner, parte do dash em que ficará as informações de dada.

```
# definição de localidade, para saída da data em português
l.setlocale(l.LC_TIME, "pt")
# data, em formato de texto, importada de datetime
data_atual = date.today()
data_em_texto = data_atual.strftime("%d de %B de %Y").title()
# banner com apresentação e data atual
top_banner = [[sg.Text('Dashboard'+ ' '*64, font='Any 20', background_color=DARK_HEADER_COLOR),
               sg.Text(data_em_texto, font='Any 20', background_color=DARK_HEADER_COLOR)]]
```

6. Previsão do tempo.

Além de definir a localidade, visando fazer a conexão com um serviço meteorológico, precisamos definir a localidade, colhemos uma chave para conexão, já informado abaixo, fazemos a declaração de um dicionário, para receber as informações requisitadas em json, depois fazemos a conversão da temperatura em graus celsius, e finalmente a configuração da área onde ficará o dash, dentro do layout.

```
# configuração da api para previsao do tempo em Brasília, através de requests
API_KEY = "7753ee82ffac2836bdb825e03be43f51"
cidade = "Brasília"
link = f"https://api.openweathermap.org/data/2.5/weather?q={cidade}&appid={API_KEY}&lang=pt_br"
requisicao = requests.get(link)
requisicao_dic = requisicao.json()
descricao = requisicao_dic['weather'][0]['description']
temperatura = requisicao_dic['main']['temp'] - 273.15
# bloco topo do dashboard com previsao do tempo
top = [[sg.Text(f"Temperatura em Brasília {temperatura:.2f} °C", size=(50,1), justification='c', pad=BPAD_TOP, font='Any 20')],
       [sg.T(f'{i*25}-{i*34}') for i in range(7)],]
```

7. Layout da mensagem de boas-vindas e gráfico.

7.1 Mensagem de boas-vinda

Criaremos um novo bloco para exibir uma mensagem de boas-vindas, que conterá um texto, para “nome do usuário” e botões para login e cancelar.

```
# bloco com input para nome do usuário, pop-up com mensagem de boas vindas
block_2 = [[sg.Text('Entrar', font='Any 20')],
           [sg.Text('\t  Pop-up', font='Any 10')],
           [sg.Input(key='-user-'), sg.Text('Nome do Usuário')],
           [sg.Button('Login'), sg.Button('Cancelar')],]
```

7.2 Gráfico

Agora um bloco para exibir um gráfico com valores aleatórios.

```
block_3 = [[sg.Text('Estatística', font='Any 20')],
           [sg.Text('\t  Gráfico', font='Any 10')],
           [sg.Image(data=sg.DEFAULT_BASE64_ICON)],
           [sg.Button('Graf'),sg.Button('Finalizar')],]  # comando para exibir imagens
```


SEPN 509 BL D 1º ANDAR ED. ISIS - ASA NORTE - BRASÍLIA/DF
☎ (61) 3340 1815 | 3447 7520 | 97401 3340
www.versatildf.com.br

8. Banco de dados.

Para podermos ter acesso ao banco de dados, precisamos dele instalado e populado em sua máquina, utilizaremos o banco de dados dbpython, conforme determinado no capítulo de configuração do banco de dados.

Criaremos alguns inputs, alguns para conexão com o bando e outros para demonstração.

```
block_4 = [[sg.Text('Cadastro', font='Any 20')],
           [sg.OptionMenu(values=('Java', 'PHP', 'Python'), default_value='Curso',  k='-OPTION MENU-'),],
           [sg.Slider(orientation='h', key='-SKIDER-')],
           [sg.Button(image_data=sg.DEFAULT_BASE64_ICON, key='-LOGO-'), sg.Text('\tInputs')],
           [sg.Input(key='-nome-'), sg.Text('Nome do Contato')],
           [sg.Input(key='-email-'), sg.Text('Endereço')],
           [sg.Input(key='-telefone-'), sg.Text('Telefone')],
           [sg.Checkbox('Cadastro', default=True, k='-CB-'),
               sg.Radio('Masc', "RadioDemo", default=True, size=(10,1), k='-R1-'), sg.Radio('Fem', "RadioDemo", default=True, size=(10,1), k='-R2-'),
               sg.Combo(values=('Lógica', 'crud', 'Web'), default_value='Módulo', readonly=True, k='-COMBO-')],
           [sg.Button('Cadastro'), sg.Button('Exit')]            ]
```

9. Finalizando o Layout.

No código a seguir temos a instrução final para construção do layout.

```
# configuração do desenho do dashboard
layout = [[sg.Column(top_banner, size=(960, 60), pad=(0,0), background_color=DARK_HEADER_COLOR)],
          [sg.Column(top, size=(920, 90), pad=BPAD_TOP)],
          [sg.Column([[sg.Column(block_2, size=(450,150), pad=BPAD_LEFT_INSIDE)],
                      [sg.Column(block_3, size=(450,150),  pad=BPAD_LEFT_INSIDE)]], pad=BPAD_LEFT, background_color=BORDER_COLOR),
           sg.Column(block_4, size=(450, 320), pad=BPAD_RIGHT)]]
# configuração da janela
window = sg.Window('Dashboard PySimpleGUI-Style', layout, margins=(0,0), background_color=BORDER_COLOR, no_titlebar=True, grab_anywhere=True)
```

10. Eventos.

Nesse ponto determinaremos a ação do botões ao serem clicados, e esse é o ponto final com a instrução window.close().

```
while True:             # evento loop
    event, values = window.read()
    if event == sg.WIN_CLOSED or event == 'Exit' or event == 'Finalizar':
        break           # parar aplicação
    elif event == 'Graf':
            grafico()   # função para rodar pop-up do gráfico
    elif event == "Login":
        sg.popup('Bem vindo ',values['-user-'], image=sg.EMOJI_BASE64_HAPPY_JOY, keep_on_top=True)   #pop-up de boas vindas
    elif event == "Cadastro":
        banco()
# fechar janela da aplicação
window.close()
```


SEPN 509 BL D 1º ANDAR ED. ISIS - ASA NORTE - BRASÍLIA/DF
(61) 3340 1815 | 3447 7520 | 97401 3340
www.versatildf.com.br